Juni Khyat
(UGC Care Group I Listed Journal)
ISSN: 2278-4632
Vol-13 Issue-01 2023

A Rounding-Based Approach to the Construction of an Effective Approximate Multiplier

Mrs. BELLAMKONDA PAVANI[1], Mr. S. CHANTI[2], Mrs. M. MAHA LAKSHMI[3]
ASSISTANT PROFESSOR[1,2,3]
DEPARTMENT OF ECE, SWARNANDHRA COLLEGE OF ENGINEERING AND TECHNOLOGY, NARASAPUR

Abstract:

Technologies including data mining, computer vision, signal and image processing, and approximate computing are well-suited to efficient data processing in error-resistant applications. The trade-off between improved circuit characteristics and the accuracy loss that results from approximative computing is application dependent. The designer of the circuit has considerable discretion in deciding how to weigh precision against other, more flexible circuit characteristics. This research gives the rounding strategy as a useful tool for handling this trade-off. To evaluate the rounding method's performance in this setting, multiplier circuits—an essential part of computation in most processors—have been considered. We can see how various rounding schemes work by comparing the output of various multipliers' circuits.

INTRODUCTION

Many contemporary electronic products, especially portable ones like tablets and smartphones, aim to minimize power usage throughout design. As low performance (speed) loss as feasible should be achieved while doing this reduction. A variety of multimedia applications rely on digital signal processing (DSP) blocks, which are essential components of these portable devices. Digital signal processing (DSP) relies heavily on multiplications in its calculations, and the arithmetic logic unit is at the core of these blocks' computation. Thus, improving CPU performance necessitates a faster multiplier with better power/energy economy. The algorithms executed by several DSP cores often provide an image or video that is fit for human consumption as the final product. So, to improve speed and energy efficiency, we may use approximations. Humans' limited visual processing skills are the root cause of all of that. Precise mathematics has many applications outside video and image processing.running the system is not dependent on functioning. The designer is free to prioritize power and energy efficiency above speed and accuracy while using approximation computing. The arithmetic units at any level of design abstraction, such as the software, logic, architecture, or circuit, may be approximated using this method. Some possible means to achieving the approximation include allowing certain timing violations (such as voltage over scaling or over clocking) and function approximation techniques (such as changing the Boolean function of a circuit). At many different design levels, methods within the domain of function approximation have been suggested for approximating arithmetic building blocks such as adders and multipliers. We provide an approximation multiplier that is efficient, uses little power, and works well with error-tolerant digital signal processors. The area-efficient and accurate recommended approximation multiplier is constructed by modifying the normal algorithm-level multiplication technique and supposing rounded input values.

Maximum Entropy

Thanks to billions of low-power devices of all kinds and large, complex commercial and research computers, we are on the verge of a data explosion. While traditional workloads like database and transaction processing continue to grow somewhat, the computational footprint of many applications that aim to derive profound insights from massive amounts of organized and unstructured data has skyrocketed. For the majority of these data types, the precision assumed by traditional computing is superfluous. Despite this, modern cognitive apps are often developed on dependable, general-purpose (and accelerated) platforms that maintain a high level of accuracy throughout. Aiming to improve processing performance within reasonable limits while reducing these constraints is the goal of approximation computing. Research in approximation computing primarily aims to determine, at each level of the system stack (from algorithms to circuits and semi-conductor devices), the maximum amount of accuracy reduction that can be achieved while still producing practical outcomes. Scientists have studied approximation computing methods, focusing mostly on enhancing a single layer of the system stack; this has yielded fruitful outcomes, such as decreased power consumption or execution times. We set out to discover in this

ISSN: 2278-4632
Vol-13 Issue-01 2023

research whether there was a cumulative benefit to applying several approximation methodologies across many layers of the system stack and if this gain was applicable to other situations. We highlighted three kinds of approximations in an example: those that consist of skipping calculations, those that approximate arithmetic operations, and those that consist of communicating between processing units. Examples chosen for examination included lowering numerical precision, delaying synchronization, and perforating a loop. The computationally demanding applications we selected may have far-reaching effects if they were inexpensive and widely used. Natural language processing, robotics, and digital signal processing were all a part of our implementations. Our research shows that we successfully perforated hot loops in all of the applications we tested by an average of 50%, which led to a commensurate reduction in overall execution time and satisfactory output quality. Cutting the data width used in the computation from the usual 64 bits to 10 or 16 bits can result in significant energy and performance savings. We discovered that by removing certain synchronization overheads from parallel applications, their execution durations might be slashed in half. Finally, our results demonstrate that these strategies work much better when used in tandem. That is, when used intelligently, the different methods do not significantly diminish the effectiveness of one another. Because approximate computing's benefits are not restricted to a small set of applications, these results call for a redesign of general-purpose processors to natively permit different kinds of approximation. Rounding is a very efficient method for packing incoming data. This method may improve speed, area, power consumption, and approximation computation, among other circuit properties. Approximate computing is very beneficial for error-resistant applications in the domains of computer vision, visual processing, pattern recognition, scientific computing, and machine learning. Multiple new lines of inquiry have emerged as a result of research in these areas within the previous decade. The multiplier is one of the computational building pieces that requires the greatest energy and time. Despite the abundance of research in this field, every solution that has been tested so far strikes a balance between accuracy and power-delay-energy. The synthesis of partial products, the reduction of partial products, and packaging are three fundamental components of a multiplier. This paper proposes an alternate method, the rounding technique, for producing a partial product from an input block. When dealing with situations where a wide range of errors might occur, the accuracy curve is an essential metric to use for management and reduction. The multiplier employs a distinct algorithm at each level. The rounding mechanism used by a 16-bit or 32-bit input block is determined by the degree of accuracy that is needed. Active or passive products are the two main categories into which the created goods fall. Compression does not need accounting for inactive partial products, because all have zero values.

MULTIPLIER APPROACH DESIGN

To perform the multiplication, the proposed approximation multiplier uses input that has been rounded off. The suggested approach rounds off the data before sending it on to the partial product manufacture. Figure 1 shows the approximation multiplier design with the proposed technique implemented. Prior to rounding the Multiplier using a rounding block, the Multiplicand is not rounded. Each input's sign bit is first stored, and the multiplication result's sign is pre-calculated based on the input signatures. Finally, the sum is given the right sign. If the multiplicative factor is a negative integer, the shape of each input block is modified to its 2's complement. An N-bit partial product is generated for every N bits of incoming data by a regular multiplier. The rounding approach, in contrast, produces incomplete products that are both active and inactive. For this purpose, the "1" coefficient Multiplier partial products are ideal. After rounding, this results in the production of a complete row of Multiplicand. Lines with no values on them represent inactive partial products. They should not be included in the reduction process because of this.

ISSN: 2278-4632
Vol-13 Issue-01 2023

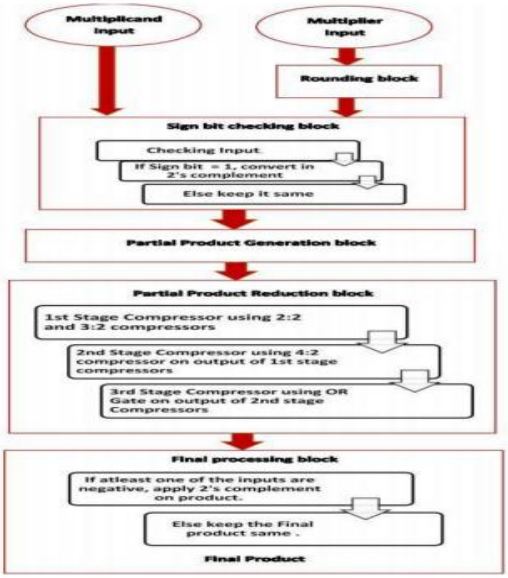

Fig. 1: 16-bit Block diagram of the

Algorithm

Careful attention to detail is required while rounding input data to avoid errors. It seems to reason that there would be less of a mistake if we rounded to smaller significant figures rather than larger ones. Consequently, the proposed approach makes use of rounding weights that are dependent on the value of the associated bit position. A rounded bit location is indistinguishable from an absolutely precise one. A rounded-off version of every exact bit is matched with it. The error gap shrinks as the bit position value increases. Figure 2's inputs "A" and "B" are summed and rounded to get the output "Br." In its simplest form, the "Rounding Method" checks the "X" bit position for a "1" and, erroneously or not, writes a "1" into the "Y" bit position.

Accurate Bit Position	Approximate Bit Position
bit0	bit1
bit1	bit1
bit2	bit1
bit3	bit4
bit4	bit4
bit5	bit4
bit6	bit7
bit7	bit7
bit8	bit7
bit9	bit10
bit10	bit10
bit11	bit10
bit12	bit13
bit13	bit13
bit14	bit15
bit15	bit15

A = 0001 0011 1000 1000
B = 0000 0011 1111 1111
Br = 0000 0100 1001 0010

15 13 10 7 4 1
Leads to active partial product rows

Fig. 2: An example after rounding „B" (16- bit)

This process continues with the "reduction of partial products," where a number of compressors are used to further compress the partial products. The suggested method gives you some discretion when it comes to reducing the number of rows that have missing product details. For instance, active partial products are condensed to just six rows in the 16-bit architecture shown in Figure 3. Partial products of size N by N are

ISSN: 2278-4632
Vol-13 Issue-01 2023

obtained by multiplying N-bit inputs, as is the case with all standard methods. As the number of bits increases, the complexity of partial products increases with O(N2). The computational cost of the suggested solution is no more than O(N6) for 16-bit values and O(N13) for 32-bit numbers. To further understand the idea and how it works, we will go through an example utilizing the A, B, and Br values given in figure 6(right) as inputs. To begin, we round up the "B" multiplier input to the "Br" number. We multiply our inputs by themselves to generate NN partial products. The NN partial products are a combination of active and inactive partial products since the input to the multiplier was rounded off. Figure 3 shows that after rounding, a multiplier with a coefficient of "1" creates a complete row of Multiplicand. So, the whole set of zeros when the multiplier coefficient was "0" is thought of as a set of inactive partial products. Therefore, it is pointless to shield them from harm while they are being reduced. In fact, it's possible that inert partial products could boost hardware performance. So, before we pack, we make sure there are no inactive parts remaining. Reduced power usage, footprint, and operating time are three ways this strategy greatly improves efficiency. To package the active partial products, three stages of compression are used. In the first stage, the partial products are compressed using whole and half adders. For 16-bit inputs, a 4:2 compressor is used for additional data compression; for 32-bit inputs, a 9:2 compressor is employed.

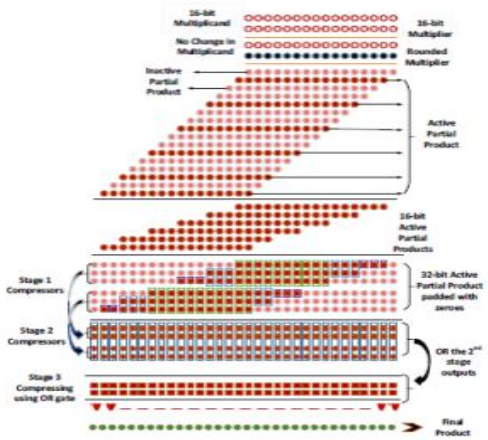

Fig. 3 (right) illustrates corresponding operations on an example.

To reduce the size of the second stage compressor's output, an OR gate is used. As an alternative to OR, complete adders are often used. It is possible to save a lot of space and electricity by using an OR gate instead of an adder altogether. Example of a multiplier (right) and its design (16-bit) are shown in Figure 3.

Model for Modified Approximate Multiplier

The basic idea behind the modified approximation multiplier is to multiply using rounded input. The partial product is computed after the data is rounded using the updated process. Prior to rounding the Multiplier using a rounding block, the Multiplicand is not rounded. Each input's sign bit is first stored, and the multiplication result's sign is pre-calculated based on the input signatures. Finally, the sum is given the right sign. A single 2n-bit Parallel Prefix adder may be used to add the updated multiplier's output, which is the absolute value of the partial products.

RESULT

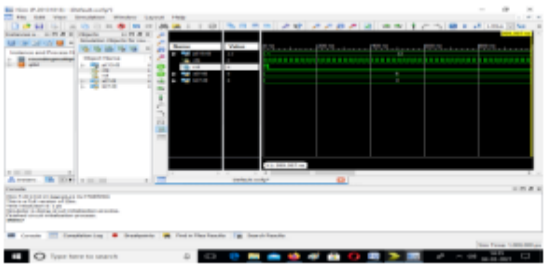

Fig.4 Output wave form

Fig 4 show the simulation result of proposed multiplier a, b, calk, rest as the input and p as the output

ISSN: 2278-4632
Vol-13 Issue-01 2023

COMPRESSION TABLE

	EXISTING ADDER	PROPOSED ADDER
DELAY	32.681ns	21.389ns
AREA	143	147
POWER	0.304w	0.186w
SPEED	30.598Mhz	46.753Mhz

CONCLUSION

Among the proposed algorithms for both signed and unsigned data, this one seems to have the best power-area latency and PDP efficiency (16-bit and 32-bit types). For the first time, this paper investigates rounding methods for an approximation multiplier with a singular emphasis on fixed active partial product rows. We can see from this rounding distribution where the chances of rounding a figure are high and low. Rounding patterns may be fine-tuned to accommodate different levels of accuracy, however it does raise the hardware cost somewhat. Whether the rounding pattern is static or dynamic, a smaller number of active partial product rows may increase compression. The proposed method has many potential uses in the fields of image processing, machine learning, and signal processing. Assigning different weights based on the bit position of "1" is important to maintain a degree of accuracy equivalent to the conventional technique. The proposed approach outperforms DRUM in terms of hardware properties due to its adaptable reduction of partial products. The refined multiplier, which got its very accurate results by rounding the inputs by 2n. By omitting the computation-intensive part of the multiplication, time and energy efficiency were improved, but a little amount of inaccuracy was sacrificed.

References

[1In his lesson titled "Ultra-low power VLSI circuit design demystified and explained," M. Alioto Published in January 2012 in the IEEE Transactions on Systems, Books I, Volume 59, Issue 1, Pages 3-29.

2 "Low-powerdigital signal processing using approximate adders" by V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy appeared in the January 2013 issue of the IEEE Transactions on Computer-Aided Design and Integration Circuits Systems, volume 32, issue 1, pages 124–137.

[3] "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," published in the April 2010 issue of the IEEE Transactions on Circuits and Systems, with authors H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas.

[4] In the Proceedings of the 2011 International Conference on Computer-Aided Design, R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan published an article titled "MACACO: Modeling and analysis of circuits for approximate computing" (pp. 667-673).

(5) "New approximate multiplier for low power digital signal processing," in Proceedings of the 17th International Symposium on Computer Architecture and Digital Systems (CADS), October 2013, pages 25-30, by F. Farshchi, M. S. Abrishami, and S. M. Fakhraie.

[6] "Trading accuracy for power with an underdesigned multiplier architecture," in Proceedings of the 24th International Conference on Very Large Scale Integration Design, January 2011, pages 346–351, by P. Kulkarni, P. Gupta, and M. Ercegovac.

The paper "Approximate signed binary integer multipliers for arithmetic data value speculation" was published in 2009 in the Proceedings of the Design, Architecture, Signal, and Image Processing Conference. It was co-authored by S. AlSarawi, B. J. Phillips, and D. R. Kelly.

The paper "Low-power high-speed multiplier for errortolerant application" was presented at the 2010 IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC) and was written by Kyaw, Goh, and Yeo (pp. 1-4).

In their 2015 April article for the IEEE Transactions on Computers, A. Momeni, J. Han, P. Montuschi, and F. Lombardi discuss the "design and analysis of approximate compressors for multiplication." Vol. 64, no. 4, pp. 984-994.

In the proceedings of the 8th International Workshop on Reconfigurable Communication-Centric Systems on Chip, K. Bhardwaj and P. S. Mane presented "ACMA: Accuracy-configurable multiplier architecture for error-resilient system-on-chip" (pp. 1-6).

[11] In paper presented at the 15th International Symposium on Quality Electronic Design (ISQED) in 2014, K. Bhardwaj, P. S. Mane, and J. Henkel discuss an approximate Wallace tree multiplier for error-resilient systems that is both power- and area-efficient (pp. 263-269).

Computer division and multiplication using binary logarithms, by J. N. Mitchell, published in IRE Transactions on Electronic Computing, volume 11, issue 4, pages 512–517, August 1962.

The paper "Improving accuracy in Mitchell"s logarithmic multiplication via operand decomposition" was published in the IEEE Transactions on Computers in December 2006 and was written by V. Mahalingam and N. Ranganathan.

ISSN: 2278-4632
Vol-13 Issue-01 2023

[14] Open Cell Library of Nangate, 45 nm, accessed in 2010. The Internet. You may access it at: http://www.nangate.com/.

The Pocket Handbook of Image Processing Algorithms in C., eds. H. R. Myler and A. R. Weeks, Prentice-Hall, 2009, Englewood Cliffs, NJ, USA.